

# Evolutionary K-means clustering (E-means) using Genetic Algorithms

1.0

Generated by Doxygen 1.8.5

Fri Aug 14 2015 03:24:52



# Contents

<b>1</b>	<b>evolutionary-clustering</b>	<b>1</b>
<b>2</b>	<b>Data Structure Index</b>	<b>3</b>
2.1	Data Structures . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Data Structure Documentation</b>	<b>7</b>
4.1	pcg_state_setseq_64 Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Field Documentation . . . . .	7
4.1.2.1	inc . . . . .	7
4.1.2.2	state . . . . .	7
<b>5</b>	<b>File Documentation</b>	<b>9</b>
5.1	include/cluster.h File Reference . . . . .	9
5.1.1	Function Documentation . . . . .	9
5.1.1.1	calc_bounds . . . . .	9
5.1.1.2	calc_centroids . . . . .	9
5.1.1.3	lloyd_defined . . . . .	10
5.1.1.4	lloyd_random . . . . .	10
5.1.1.5	random_centroids . . . . .	11
5.2	include/fitness.h File Reference . . . . .	11
5.2.1	Function Documentation . . . . .	12
5.2.1.1	dunn_index . . . . .	12
5.3	include/io.h File Reference . . . . .	12
5.3.1	Function Documentation . . . . .	12
5.3.1.1	load_data . . . . .	12
5.3.1.2	save_results . . . . .	12
5.4	include/operators.h File Reference . . . . .	13
5.4.1	Function Documentation . . . . .	13
5.4.1.1	crossover . . . . .	13

5.4.1.2	mutate	14
5.5	include/pcg_basic.h File Reference	15
5.5.1	Macro Definition Documentation	15
5.5.1.1	PCG32_INITIALIZER	15
5.5.2	Typedef Documentation	16
5.5.2.1	pcg32_random_t	16
5.5.3	Function Documentation	16
5.5.3.1	pcg32_boundedrand	16
5.5.3.2	pcg32_boundedrand_r	16
5.5.3.3	pcg32_random	16
5.5.3.4	pcg32_random_r	17
5.5.3.5	pcg32_srandom	17
5.5.3.6	pcg32_srandom_r	17
5.6	include/selection.h File Reference	17
5.6.1	Function Documentation	17
5.6.1.1	gen_probability	17
5.6.1.2	select_parent	18
5.7	include/utility.h File Reference	18
5.7.1	Macro Definition Documentation	19
5.7.1.1	BLUE	19
5.7.1.2	CYAN	19
5.7.1.3	GREEN	19
5.7.1.4	MAGENTA	19
5.7.1.5	RED	19
5.7.1.6	RESET	19
5.7.1.7	YELLOW	19
5.7.2	Enumeration Type Documentation	19
5.7.2.1	debug_code	19
5.7.2.2	error_code	20
5.7.3	Variable Documentation	20
5.7.3.1	DEBUG	20
5.7.3.2	VERBOSE	20
5.8	README.md File Reference	20
5.9	src/cluster.c File Reference	20
5.9.1	Function Documentation	21
5.9.1.1	calc_bounds	21
5.9.1.2	calc_centroids	21
5.9.1.3	lloyd_defined	21
5.9.1.4	lloyd_random	22
5.9.1.5	random_centroids	22

5.10	src/emeans.c File Reference	23
5.10.1	Function Documentation	24
5.10.1.1	emeans	24
5.10.1.2	main	25
5.10.2	Variable Documentation	25
5.10.2.1	c_rate	25
5.10.2.2	centroids_file	25
5.10.2.3	cfg	25
5.10.2.4	cluster_file	25
5.10.2.5	data_cols	25
5.10.2.6	data_file	26
5.10.2.7	data_rows	26
5.10.2.8	DEBUG	26
5.10.2.9	fitness_file	26
5.10.2.10	m_rate	26
5.10.2.11	max_iter	26
5.10.2.12	n_clusters	26
5.10.2.13	opts	26
5.10.2.14	size	26
5.10.2.15	trials	26
5.10.2.16	VERBOSE	27
5.11	src/fitness.c File Reference	27
5.11.1	Function Documentation	27
5.11.1.1	dunn_index	27
5.12	src/io.c File Reference	27
5.12.1	Function Documentation	28
5.12.1.1	load_data	28
5.12.1.2	save_results	28
5.13	src/operators.c File Reference	29
5.13.1	Function Documentation	29
5.13.1.1	crossover	29
5.13.1.2	mutate	30
5.14	src/pcg_basic.c File Reference	31
5.14.1	Function Documentation	31
5.14.1.1	pcg32_boundedrand	31
5.14.1.2	pcg32_boundedrand_r	31
5.14.1.3	pcg32_random	32
5.14.1.4	pcg32_random_r	32
5.14.1.5	pcg32_srandom	32
5.14.1.6	pcg32_srandom_r	32

5.15	src/selection.c File Reference . . . . .	33
5.15.1	Function Documentation . . . . .	33
5.15.1.1	gen_probability . . . . .	33
5.15.1.2	select_parent . . . . .	33
<b>Index</b>		<b>35</b>

## **Chapter 1**

# **evolutionary-clustering**

An enhanced highly parallel K-means clustering algorithm using evolutionary strategies to perform metaheuristic optimization.





## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">pcg_state_setseq_64</a> . . . . .	7
---	---



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

include/ <a href="#">cluster.h</a> . . . . .	9
include/ <a href="#">fitness.h</a> . . . . .	11
include/ <a href="#">io.h</a> . . . . .	12
include/ <a href="#">operators.h</a> . . . . .	13
include/ <a href="#">pcg_basic.h</a> . . . . .	15
include/ <a href="#">selection.h</a> . . . . .	17
include/ <a href="#">utility.h</a> . . . . .	18
src/ <a href="#">cluster.c</a> . . . . .	20
src/ <a href="#">emeans.c</a> . . . . .	23
src/ <a href="#">fitness.c</a> . . . . .	27
src/ <a href="#">io.c</a> . . . . .	27
src/ <a href="#">operators.c</a> . . . . .	29
src/ <a href="#">pcg_basic.c</a> . . . . .	31
src/ <a href="#">selection.c</a> . . . . .	33



## Chapter 4

# Data Structure Documentation

### 4.1 pcg\_state\_setseq\_64 Struct Reference

```
#include <pcg_basic.h>
```

#### Data Fields

- uint64\_t [state](#)
- uint64\_t [inc](#)

#### 4.1.1 Detailed Description

Definition at line 40 of file pcg\_basic.h.

#### 4.1.2 Field Documentation

##### 4.1.2.1 uint64\_t inc

Definition at line 42 of file pcg\_basic.h.

##### 4.1.2.2 uint64\_t state

Definition at line 41 of file pcg\_basic.h.

The documentation for this struct was generated from the following file:

- [include/pcg\\_basic.h](#)



## Chapter 5

# File Documentation

### 5.1 include/cluster.h File Reference

```
#include <gsl/gsl_matrix.h>
#include "pcg_basic.h"
```

#### Functions

- int [lloyd\\_random](#) (int [trials](#), gsl\_matrix \*data, int [n\\_clusters](#), gsl\_matrix \*\*clusters, [pcg32\\_random\\_t](#) \*rng)
- int [lloyd\\_defined](#) (int [trials](#), gsl\_matrix \*centroids, gsl\_matrix \*data, int [n\\_clusters](#), gsl\_matrix \*\*clusters)
- int [calc\\_centroids](#) (gsl\_matrix \*centroids, gsl\_matrix \*data, int [n\\_clusters](#), gsl\_matrix \*\*clusters)
- int [calc\\_bounds](#) (gsl\_matrix \*data, gsl\_matrix \*bounds)
- int [random\\_centroids](#) (gsl\_matrix \*centroids, gsl\_matrix \*bounds, [pcg32\\_random\\_t](#) \*rng)

#### 5.1.1 Function Documentation

##### 5.1.1.1 int [calc\\_bounds](#) ( [gsl\\_matrix](#) \* *data*, [gsl\\_matrix](#) \* *bounds* )

Calculates the minimum and maximum bounds based on the data.

##### Parameters

<i>data</i>	Point to matrix containing the data
<i>bounds</i>	The min/max bounds for each dimensions of the data

##### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 334 of file cluster.c.

##### 5.1.1.2 int [calc\\_centroids](#) ( [gsl\\_matrix](#) \* *centroids*, [gsl\\_matrix](#) \* *data*, int *n\_clusters*, [gsl\\_matrix](#) \*\* *clusters* )

Calculate the new centroids using the clustering assignment.

**Parameters**

<i>centroids</i>	Pointer to matrix containing centroids to be updated
<i>data</i>	Point to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>cluster</i>	Pointer to vector containing cluster assignment

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 307 of file cluster.c.

5.1.1.3 `int lloyd_defined ( int trials, gsl_matrix * centroids, gsl_matrix * data, int n_clusters, gsl_matrix ** clusters )`

Performs Lloyd's algorithm using the defined centroids.

**Parameters**

<i>trials</i>	Number of trials to perform
<i>centroids</i>	Pointer to matrix containing the centroids
<i>data</i>	Pointer to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 191 of file cluster.c.

Here is the call graph for this function:



5.1.1.4 `int lloyd_random ( int trials, gsl_matrix * data, int n_clusters, gsl_matrix ** clusters, pcg32_random_t * rng )`

Performs Lloyd's algorithm using random initial centroids.

**Parameters**

<i>trials</i>	Number of trials to perform
<i>data</i>	Pointer to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters



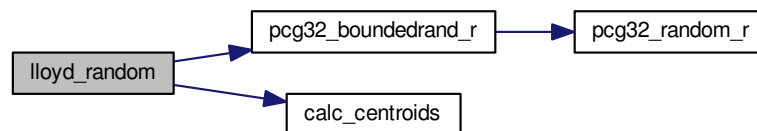
<i>rng</i>	Pointer to the random number generator
------------	--

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 33 of file cluster.c.

Here is the call graph for this function:



#### 5.1.1.5 int random\_centroids ( gsl\_matrix \* *centroids*, gsl\_matrix \* *bounds*, pcg32\_random\_t \* *rng* )

Generates random centroids within the bounds for each dimension.

**Parameters**

<i>centroids</i>	Pointer to matrix containing centroids to be updated
<i>bounds</i>	The min/max bounds for each dimensions of the data
<i>rng</i>	Pointer to the random number generator

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 362 of file cluster.c.

Here is the call graph for this function:



## 5.2 include/fitness.h File Reference

```
#include <gsl/gsl_matrix.h>
```

## Functions

- double [dunn\\_index](#) (gsl\_matrix \*centroids, int [n\\_clusters](#), gsl\_matrix \*\*clusters)

### 5.2.1 Function Documentation

#### 5.2.1.1 double dunn\_index ( gsl\_matrix \* *centroids*, int *n\_clusters*, gsl\_matrix \*\* *clusters* )

Calculates the Dunn Index, a metric for evaluating the clustering results.

##### Parameters

<i>centroids</i>	Pointer to matrix containing the centroids
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters

##### Returns

The Dunn Index

Definition at line 32 of file fitness.c.

## 5.3 include/io.h File Reference

```
#include <gsl/gsl_matrix.h>
```

## Functions

- int [load\\_data](#) (char \*input, gsl\_matrix \*data)
- int [save\\_results](#) (char \*output, char \*output2, char \*output3, int [size](#), double fitness[[size](#)], gsl\_matrix \*\*population, int [n\\_clusters](#), gsl\_matrix \*\*\*clusters)

### 5.3.1 Function Documentation

#### 5.3.1.1 int load\_data ( char \* *input*, gsl\_matrix \* *data* )

Loads the data from as CSV file into a matrix,

##### Parameters

<i>input</i>	Path to the data file
<i>data</i>	Pointer to the GSL matrix to be populated

##### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 123 of file io.c.

#### 5.3.1.2 int save\_results ( char \* *output*, char \* *output2*, char \* *output3*, int *size*, double fitness[*size*], gsl\_matrix \*\* *population*, int *n\_clusters*, gsl\_matrix \*\*\* *clusters* )

Save the chromosome and fitness value if they are better than previous.

## Parameters

<i>output</i>	Path to save the optimal fitness value
<i>output2</i>	Path to save the optimal fitness centroids
<i>output3</i>	Path to save the optimal cluster results
<i>size</i>	Size of the populations
<i>fitness</i>	Pointer to array of fitness values for the population
<i>population</i>	Population of all chromosomes
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	The clusters for each chromosome in the population

## Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 28 of file io.c.

## 5.4 include/operators.h File Reference

```
#include <gsl/gsl_matrix.h>
#include "pcg_basic.h"
```

## Functions

- void [crossover](#) (gsl\_matrix \*parent1, gsl\_matrix \*parent2, [pcg32\\_random\\_t](#) \*rng)
- void [mutate](#) (gsl\_matrix \*chromosome, gsl\_matrix \*bounds, [pcg32\\_random\\_t](#) \*rng)

### 5.4.1 Function Documentation

#### 5.4.1.1 void crossover ( gsl\_matrix \* parent1, gsl\_matrix \* parent2, pcg32\_random\_t \* rng )

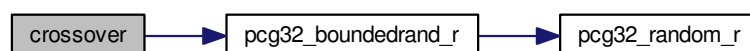
Performs chromosome crossover by randomly selecting a crossover point and randomly either swapping the top or the bottom half.

## Parameters

<i>len</i>	The length of the chromosome
<i>parent1</i>	The first parent chromosome
<i>parent2</i>	The second parent chromosome
<i>rng</i>	Pointer to the random number generator

Definition at line 34 of file operators.c.

Here is the call graph for this function:



5.4.1.2 void mutate ( gsl\_matrix \* *chromosome*, gsl\_matrix \* *bounds*, pcg32\_random\_t \* *rng* )

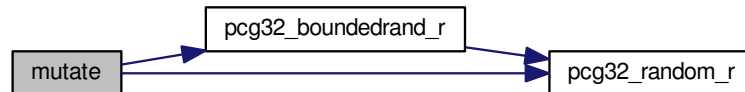
Performs mutation, selects a random row and column in the chromosome and mutates it to a random value within the min/max bounds.

## Parameters

<i>chromosome</i>	The chromosome
<i>bounds</i>	The min/max bounds for each dimensions of the data
<i>rng</i>	Pointer to the random number generator

Definition at line 106 of file operators.c.

Here is the call graph for this function:



## 5.5 include/pcg\_basic.h File Reference

```
#include <inttypes.h>
```

### Data Structures

- struct [pcg\\_state\\_setseq\\_64](#)

### Macros

- #define [PCG32\\_INITIALIZER](#) { 0x853c49e6748fea9bULL, 0xda3e39cb94b95bdbULL }

### Typedefs

- typedef struct [pcg\\_state\\_setseq\\_64](#) [pcg32\\_random\\_t](#)

### Functions

- void [pcg32\\_srandom](#) (uint64\_t initstate, uint64\_t initseq)
- void [pcg32\\_srandom\\_r](#) ([pcg32\\_random\\_t](#) \*rng, uint64\_t initstate, uint64\_t initseq)
- uint32\_t [pcg32\\_random](#) (void)
- uint32\_t [pcg32\\_random\\_r](#) ([pcg32\\_random\\_t](#) \*rng)
- uint32\_t [pcg32\\_boundedrand](#) (uint32\_t bound)
- uint32\_t [pcg32\\_boundedrand\\_r](#) ([pcg32\\_random\\_t](#) \*rng, uint32\_t bound)

#### 5.5.1 Macro Definition Documentation

##### 5.5.1.1 #define PCG32\_INITIALIZER { 0x853c49e6748fea9bULL, 0xda3e39cb94b95bdbULL }

Definition at line 49 of file `pcg_basic.h`.

## 5.5.2 Typedef Documentation

### 5.5.2.1 typedef struct pcg\_state\_setseq\_64 pcg32\_random\_t

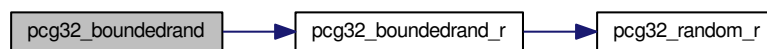
Definition at line 45 of file pcg\_basic.h.

## 5.5.3 Function Documentation

### 5.5.3.1 uint32\_t pcg32\_boundedrand ( uint32\_t bound )

Definition at line 112 of file pcg\_basic.c.

Here is the call graph for this function:



### 5.5.3.2 uint32\_t pcg32\_boundedrand\_r ( pcg32\_random\_t \* rng, uint32\_t bound )

Definition at line 79 of file pcg\_basic.c.

Here is the call graph for this function:



### 5.5.3.3 uint32\_t pcg32\_random ( void )

Definition at line 69 of file pcg\_basic.c.

Here is the call graph for this function:



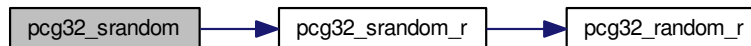
#### 5.5.3.4 uint32\_t pcg32\_random\_r ( pcg32\_random\_t \* rng )

Definition at line 60 of file pcg\_basic.c.

#### 5.5.3.5 void pcg32\_srandom ( uint64\_t *initstate*, uint64\_t *initseq* )

Definition at line 51 of file pcg\_basic.c.

Here is the call graph for this function:



#### 5.5.3.6 void pcg32\_srandom\_r ( pcg32\_random\_t \* rng, uint64\_t *initstate*, uint64\_t *initseq* )

Definition at line 42 of file pcg\_basic.c.

Here is the call graph for this function:



## 5.6 include/selection.h File Reference

```
#include "pcg_basic.h"
```

### Functions

- void [gen\\_probability](#) (int [size](#), double *fitness[size]*, double *probability[size]*)
- int [select\\_parent](#) (int [size](#), double *probability[size]*, [pcg32\\_random\\_t](#) \*rng)

#### 5.6.1 Function Documentation

##### 5.6.1.1 void gen\_probability ( int *size*, double *fitness[size]*, double *probability[size]* )

Perform the roulette wheel probability selection, an array is populated with the index of the chromosome to select with a frequency based on the fitness value.

## Parameters

<i>size</i>	The size of the population
<i>fitness</i>	Pointer to an array of fitness values for population
<i>probability</i>	Pointer to the probability array, populated by function

Definition at line 29 of file selection.c.

#### 5.6.1.2 int select\_parent ( int size, double probability[size], pcg32\_random\_t \* rng )

Selects a parent from the population at random with a probability of being selected based on the probabilities provided.

## Parameters

<i>size</i>	The size of the population
<i>probability</i>	Probabilities of each chromosome in population being selected
<i>rng</i>	Pointer to the random number generator

## Returns

The index of the parent in the population to select

Definition at line 67 of file selection.c.

Here is the call graph for this function:



## 5.7 include/utility.h File Reference

## Macros

- `#define RED "\x1b[31m"`
- `#define GREEN "\x1b[32m"`
- `#define YELLOW "\x1b[33m"`
- `#define BLUE "\x1b[34m"`
- `#define MAGENTA "\x1b[35m"`
- `#define CYAN "\x1b[36m"`
- `#define RESET "\x1b[0m"`

## Enumerations

- `enum debug_code {`  
`DEBUG_CONFIG = 1, DEBUG_DATA = 2, DEBUG_CLUSTER = 3, DEBUG_BOUNDS = 4,`  
`DEBUG_CENTROIDS = 5, DEBUG_DUNN = 6, DEBUG_CROSSOVER = 7, DEBUG_MUTATE = 8,`  
`DEBUG_PROBABILITY = 9 }`

*Enumeration of the DEBUG codes.*



- enum `error_code` { `SUCCESS` = 0, `ERROR` = 1 }
- Error codes.*

## Variables

- int `DEBUG`
- int `VERBOSE`

### 5.7.1 Macro Definition Documentation

#### 5.7.1.1 `#define BLUE "\x1b[34m"`

Definition at line 27 of file utility.h.

#### 5.7.1.2 `#define CYAN "\x1b[36m"`

Definition at line 29 of file utility.h.

#### 5.7.1.3 `#define GREEN "\x1b[32m"`

Definition at line 25 of file utility.h.

#### 5.7.1.4 `#define MAGENTA "\x1b[35m"`

Definition at line 28 of file utility.h.

#### 5.7.1.5 `#define RED "\x1b[31m"`

Definition at line 24 of file utility.h.

#### 5.7.1.6 `#define RESET "\x1b[0m"`

Definition at line 30 of file utility.h.

#### 5.7.1.7 `#define YELLOW "\x1b[33m"`

Definition at line 26 of file utility.h.

### 5.7.2 Enumeration Type Documentation

#### 5.7.2.1 enum `debug_code`

Enumeration of the DEBUG codes.

## Enumerator

**`DEBUG_CONFIG`** Print all the values parsed from the config file

**`DEBUG_DATA`** Print the contents of the data file

**`DEBUG_CLUSTER`** Debug the clustering process using Lloyd's

**`DEBUG_BOUNDS`** Debug the min/max bounds for each dimension

**DEBUG\_CENTROIDS** Debug the randomly generated initial centroids  
**DEBUG\_DUNN** Debug the Dunn Index calculations  
**DEBUG\_CROSSOVER** Debug the crossover operator  
**DEBUG\_MUTATE** Debug the mutation operator  
**DEBUG\_PROBABILITY** Debug output for the probability generation

Definition at line 39 of file utility.h.

#### 5.7.2.2 enum error\_code

Error codes.

Enumerator

**SUCCESS** Successful execution  
**ERROR** Generic error code

Definition at line 56 of file utility.h.

### 5.7.3 Variable Documentation

#### 5.7.3.1 int DEBUG

Definition at line 40 of file emeans.c.

#### 5.7.3.2 int VERBOSE

Definition at line 40 of file emeans.c.

## 5.8 README.md File Reference

## 5.9 src/cluster.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "pcg_basic.h"
#include "cluster.h"
```

### Functions

- [int lloyd\\_random](#) (int trials, gsl\_matrix \*data, int n\_clusters, gsl\_matrix \*\*clusters, [pcg32\\_random\\_t](#) \*rng)
- [int lloyd\\_defined](#) (int trials, gsl\_matrix \*centroids, gsl\_matrix \*data, int n\_clusters, gsl\_matrix \*\*clusters)

- int [calc\\_centroids](#) (gsl\_matrix \*centroids, gsl\_matrix \*data, int [n\\_clusters](#), gsl\_matrix \*\*clusters)
- int [calc\\_bounds](#) (gsl\_matrix \*data, gsl\_matrix \*bounds)
- int [random\\_centroids](#) (gsl\_matrix \*centroids, gsl\_matrix \*bounds, [pcg32\\_random\\_t](#) \*rng)

## 5.9.1 Function Documentation

### 5.9.1.1 int [calc\\_bounds](#) ( [gsl\\_matrix](#) \* *data*, [gsl\\_matrix](#) \* *bounds* )

Calculates the minimum and maximum bounds based on the data.

#### Parameters

<i>data</i>	Point to matrix containing the data
<i>bounds</i>	The min/max bounds for each dimensions of the data

#### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 334 of file cluster.c.

### 5.9.1.2 int [calc\\_centroids](#) ( [gsl\\_matrix](#) \* *centroids*, [gsl\\_matrix](#) \* *data*, int *n\_clusters*, [gsl\\_matrix](#) \*\* *clusters* )

Calculate the new centroids using the clustering assignment.

#### Parameters

<i>centroids</i>	Pointer to matrix containing centroids to be updated
<i>data</i>	Point to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>cluster</i>	Pointer to vector containing cluster assignment

#### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 307 of file cluster.c.

### 5.9.1.3 int [lloyd\\_defined](#) ( int *trials*, [gsl\\_matrix](#) \* *centroids*, [gsl\\_matrix](#) \* *data*, int *n\_clusters*, [gsl\\_matrix](#) \*\* *clusters* )

Performs Lloyd's algorithm using the defined centroids.

#### Parameters

<i>trials</i>	Number of trials to perform
<i>centroids</i>	Pointer to matrix containing the centroids
<i>data</i>	Pointer to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters

#### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 191 of file cluster.c.

Here is the call graph for this function:



#### 5.9.1.4 int lloyd\_random ( int *trials*, gsl\_matrix \* *data*, int *n\_clusters*, gsl\_matrix \*\* *clusters*, pcg32\_random\_t \* *rng* )

Performs Lloyd's algorithm using random initial centroids.

##### Parameters

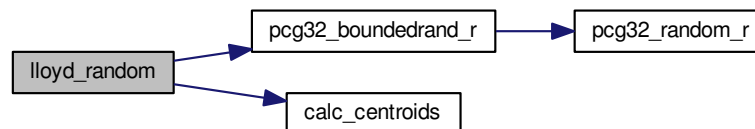
<i>trials</i>	Number of trials to perform
<i>data</i>	Pointer to matrix containing the data
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters
<i>rng</i>	Pointer to the random number generator

##### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 33 of file cluster.c.

Here is the call graph for this function:



#### 5.9.1.5 int random\_centroids ( gsl\_matrix \* *centroids*, gsl\_matrix \* *bounds*, pcg32\_random\_t \* *rng* )

Generates random centroids within the bounds for each dimension.

##### Parameters

<i>centroids</i>	Pointer to matrix containing centroids to be updated
<i>bounds</i>	The min/max bounds for each dimensions of the data

<i>rng</i>	Pointer to the random number generator
------------	--

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 362 of file cluster.c.

Here is the call graph for this function:

**5.10 src/emeans.c File Reference**

```

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <time.h>
#include <confuse.h>
#include <gsl/gsl_matrix.h>
#include "utility.h"
#include "pcg_basic.h"
#include "io.h"
#include "cluster.h"
#include "fitness.h"
#include "operators.h"
#include "selection.h"

```

**Functions**

- int [emeans](#) (void)
- int [main](#) (int argc, char \*argv[])

**Variables**

- int [DEBUG](#)
- int [VERBOSE](#)
- int64\_t [n\\_clusters](#) = 3
- int64\_t [trials](#) = 1
- int64\_t [size](#) = 100
- double [m\\_rate](#) = 0.01

- double `c_rate` = 0.70
- int64\_t `max_iter` = 10000
- int64\_t `data_rows` = 0
- int64\_t `data_cols` = 0
- char \* `data_file` = NULL
- char \* `centroids_file` = NULL
- char \* `fitness_file` = NULL
- char \* `cluster_file` = NULL
- cfg\_opt\_t `opts` []
- cfg\_t \* `cfg`

## 5.10.1 Function Documentation

### 5.10.1.1 int emeans ( void )

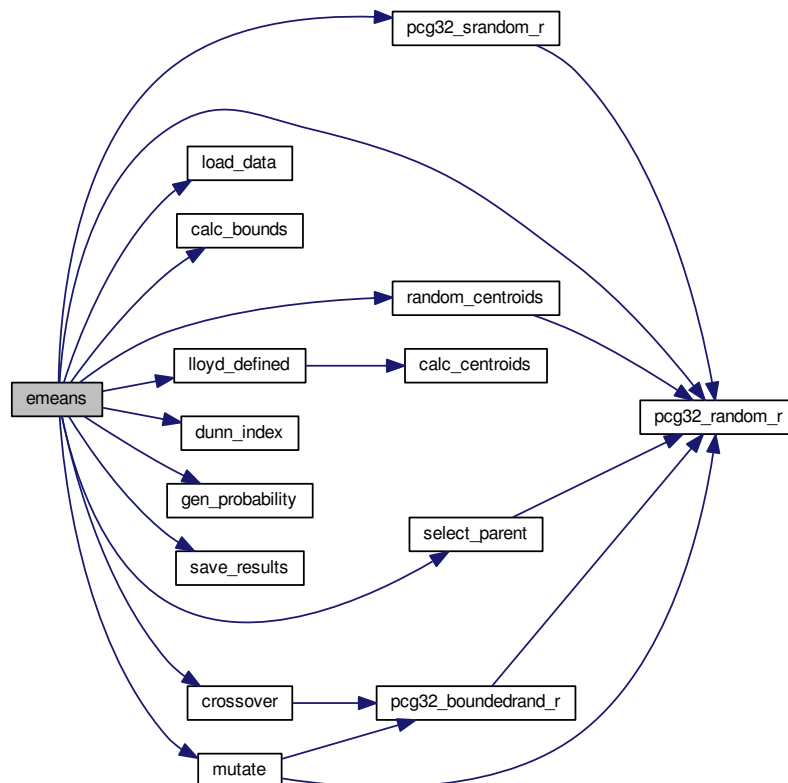
The E-means algorithm, uses a genetic algorithm to optimize the parameters for the K-means implementation of clustering based Lloyds clustering algorithm.

#### Returns

Status code, 0 for SUCCESS, 1 for ERROR

Definition at line 81 of file emeans.c.

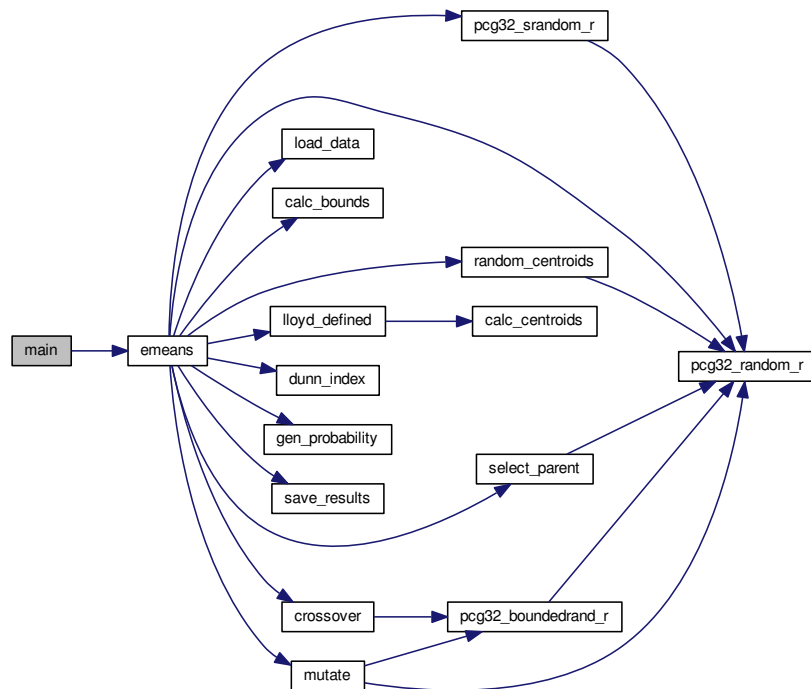
Here is the call graph for this function:



### 5.10.1.2 `int main ( int argc, char * argv[] )`

Definition at line 241 of file emeans.c.

Here is the call graph for this function:



## 5.10.2 Variable Documentation

### 5.10.2.1 `double c_rate = 0.70`

Definition at line 47 of file emeans.c.

### 5.10.2.2 `char * centroids_file = NULL`

Definition at line 52 of file emeans.c.

### 5.10.2.3 `cfg_t* cfg`

Definition at line 72 of file emeans.c.

### 5.10.2.4 `char * cluster_file = NULL`

Definition at line 54 of file emeans.c.

### 5.10.2.5 `int64_t data_cols = 0`

Definition at line 50 of file emeans.c.

#### 5.10.2.6 `char* data_file = NULL`

Definition at line 51 of file `emeans.c`.

#### 5.10.2.7 `int64_t data_rows = 0`

Definition at line 49 of file `emeans.c`.

#### 5.10.2.8 `int DEBUG`

Definition at line 40 of file `emeans.c`.

#### 5.10.2.9 `char * fitness_file = NULL`

Definition at line 53 of file `emeans.c`.

#### 5.10.2.10 `double m_rate = 0.01`

Definition at line 46 of file `emeans.c`.

#### 5.10.2.11 `int64_t max_iter = 10000`

Definition at line 48 of file `emeans.c`.

#### 5.10.2.12 `int64_t n_clusters = 3`

Definition at line 43 of file `emeans.c`.

#### 5.10.2.13 `cfg_opt_t opts[]`

**Initial value:**

```
= {
    CFG_SIMPLE_INT("n_clusters", &n_clusters),
    CFG_SIMPLE_INT("trials", &trials),
    CFG_SIMPLE_INT("size", &size),
    CFG_SIMPLE_FLOAT("m_rate", &m_rate),
    CFG_SIMPLE_FLOAT("c_rate", &c_rate),
    CFG_SIMPLE_INT("max_iter", &max_iter),
    CFG_SIMPLE_INT("data_rows", &data_rows),
    CFG_SIMPLE_INT("data_cols", &data_cols),
    CFG_SIMPLE_STR("data_file", &data_file),
    CFG_SIMPLE_STR("centroids_file", &centroids_file),
    CFG_SIMPLE_STR("fitness_file", &fitness_file),
    CFG_SIMPLE_STR("cluster_file", &cluster_file),
}
```

Definition at line 57 of file `emeans.c`.

#### 5.10.2.14 `int64_t size = 100`

Definition at line 45 of file `emeans.c`.

#### 5.10.2.15 `int64_t trials = 1`

Definition at line 44 of file `emeans.c`.



## 5.10.2.16 int VERBOSE

Definition at line 40 of file emeans.c.

## 5.11 src/fitness.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "fitness.h"
```

### Functions

- double [dunn\\_index](#) (gsl\_matrix \*centroids, int [n\\_clusters](#), gsl\_matrix \*\*clusters)

#### 5.11.1 Function Documentation

5.11.1.1 double [dunn\\_index](#) ( gsl\_matrix \* *centroids*, int *n\_clusters*, gsl\_matrix \*\* *clusters* )

Calculates the Dunn Index, a metric for evaluating the clustering results.

##### Parameters

<i>centroids</i>	Pointer to matrix containing the centroids
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	Pointer to array of matrices containing data in clusters

##### Returns

The Dunn Index

Definition at line 32 of file fitness.c.

## 5.12 src/io.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <float.h>
#include "utility.h"
#include "io.h"
```

## Functions

- int [save\\_results](#) (char \*output, char \*output2, char \*output3, int [size](#), double fitness[[size](#)], gsl\_matrix \*\*population, int [n\\_clusters](#), gsl\_matrix \*\*\*clusters)
- int [load\\_data](#) (char \*input, gsl\_matrix \*data)

### 5.12.1 Function Documentation

#### 5.12.1.1 int load\_data ( char \* *input*, gsl\_matrix \* *data* )

Loads the data from as CSV file into a matrix,

##### Parameters

<i>input</i>	Path to the data file
<i>data</i>	Pointer to the GSL matrix to be populated

##### Returns

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 123 of file io.c.

#### 5.12.1.2 int save\_results ( char \* *output*, char \* *output2*, char \* *output3*, int *size*, double *fitness*[*size*], gsl\_matrix \*\* *population*, int *n\_clusters*, gsl\_matrix \*\*\* *clusters* )

Save the chromosome and fitness value if they are better than previous.

##### Parameters

<i>output</i>	Path to save the optimal fitness value
<i>output2</i>	Path to save the optimal fitness centroids
<i>output3</i>	Path to save the optimal cluster results
<i>size</i>	Size of the populations
<i>fitness</i>	Pointer to array of fitness values for the population
<i>population</i>	Population of all chromosomes
<i>n_clusters</i>	The number of clusters
<i>clusters</i>	The clusters for each chromosome in the population

**Returns**

The status code, 0 for SUCCESS, 1 for ERROR

Definition at line 28 of file io.c.

**5.13 src/operators.c File Reference**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_statistics.h>
#include "utility.h"
#include "fitness.h"
#include "operators.h"
```

**Functions**

- void [crossover](#) (gsl\_matrix \*parent1, gsl\_matrix \*parent2, [pcg32\\_random\\_t](#) \*rng)
- void [mutate](#) (gsl\_matrix \*chromosome, gsl\_matrix \*bounds, [pcg32\\_random\\_t](#) \*rng)

**5.13.1 Function Documentation****5.13.1.1 void crossover ( gsl\_matrix \* parent1, gsl\_matrix \* parent2, pcg32\_random\_t \* rng )**

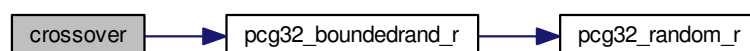
Performs chromosome crossover by randomly selecting a crossover point and randomly either swapping the top or the bottom half.

**Parameters**

<i>len</i>	The length of the chromosome
<i>parent1</i>	The first parent chromosome
<i>parent2</i>	The second parent chromosome
<i>rng</i>	Pointer to the random number generator

Definition at line 34 of file operators.c.

Here is the call graph for this function:



5.13.1.2 `void mutate ( gsl_matrix * chromosome, gsl_matrix * bounds, pcg32_random_t * rng )`

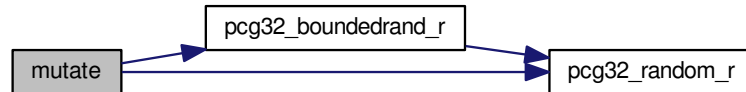
Performs mutation, selects a random row and column in the chromosome and mutates it to a random value within the min/max bounds.

## Parameters

<i>chromosome</i>	The chromosome
<i>bounds</i>	The min/max bounds for each dimensions of the data
<i>rng</i>	Pointer to the random number generator

Definition at line 106 of file operators.c.

Here is the call graph for this function:



## 5.14 src/pcg\_basic.c File Reference

```
#include "pcg_basic.h"
```

## Functions

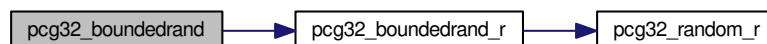
- void [pcg32\\_srandom\\_r](#) ([pcg32\\_random\\_t](#) \*rng, uint64\_t initstate, uint64\_t initseq)
- void [pcg32\\_srandom](#) (uint64\_t seed, uint64\_t seq)
- uint32\_t [pcg32\\_random\\_r](#) ([pcg32\\_random\\_t](#) \*rng)
- uint32\_t [pcg32\\_random](#) ()
- uint32\_t [pcg32\\_boundedrand\\_r](#) ([pcg32\\_random\\_t](#) \*rng, uint32\_t bound)
- uint32\_t [pcg32\\_boundedrand](#) (uint32\_t bound)

## 5.14.1 Function Documentation

## 5.14.1.1 uint32\_t pcg32\_boundedrand ( uint32\_t bound )

Definition at line 112 of file pcg\_basic.c.

Here is the call graph for this function:



## 5.14.1.2 uint32\_t pcg32\_boundedrand\_r ( pcg32\_random\_t \* rng, uint32\_t bound )

Definition at line 79 of file pcg\_basic.c.

Here is the call graph for this function:



#### 5.14.1.3 `uint32_t pcg32_random ( void )`

Definition at line 69 of file `pcg_basic.c`.

Here is the call graph for this function:



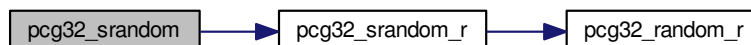
#### 5.14.1.4 `uint32_t pcg32_random_r ( pcg32_random_t * rng )`

Definition at line 60 of file `pcg_basic.c`.

#### 5.14.1.5 `void pcg32_srandom ( uint64_t seed, uint64_t seq )`

Definition at line 51 of file `pcg_basic.c`.

Here is the call graph for this function:



#### 5.14.1.6 `void pcg32_srandom_r ( pcg32_random_t * rng, uint64_t initstate, uint64_t initseq )`

Definition at line 42 of file `pcg_basic.c`.

Here is the call graph for this function:



## 5.15 src/selection.c File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <stdbool.h>
#include "pcg_basic.h"
#include "utility.h"
#include "selection.h"

```

### Functions

- void [gen\\_probability](#) (int [size](#), double fitness[[size](#)], double probability[[size](#)])
- int [select\\_parent](#) (int [size](#), double probability[[size](#)], [pcg32\\_random\\_t](#) \*rng)

#### 5.15.1 Function Documentation

##### 5.15.1.1 void [gen\\_probability](#) ( int *size*, double *fitness[size]*, double *probability[size]* )

Perform the roulette wheel probability selection, an array is populated with the index of the chromosome to select with a frequency based on the fitness value.

###### Parameters

<i>size</i>	The size of the population
<i>fitness</i>	Pointer to an array of fitness values for population
<i>probability</i>	Pointer to the probability array, populated by function

Definition at line 29 of file selection.c.

##### 5.15.1.2 int [select\\_parent](#) ( int *size*, double *probability[size]*, [pcg32\\_random\\_t](#) \* *rng* )

Selects a parent from the population at random with a probability of being selected based on the probabilities provided.

###### Parameters

<i>size</i>	The size of the population
<i>probability</i>	Probabilities of each chromosome in population being selected
<i>rng</i>	Pointer to the random number generator

#### Returns

The index of the parent in the population to select

Definition at line 67 of file selection.c.

Here is the call graph for this function:





# Index

BLUE  
    utility.h, 19

c\_rate  
    emeans.c, 25

CYAN  
    utility.h, 19

calc\_bounds  
    cluster.c, 21  
    cluster.h, 9

calc\_centroids  
    cluster.c, 21  
    cluster.h, 9

centroids\_file  
    emeans.c, 25

cfg  
    emeans.c, 25

cluster.c  
    calc\_bounds, 21  
    calc\_centroids, 21  
    lloyd\_defined, 21  
    lloyd\_random, 22  
    random\_centroids, 22

cluster.h  
    calc\_bounds, 9  
    calc\_centroids, 9  
    lloyd\_defined, 10  
    lloyd\_random, 10  
    random\_centroids, 11

cluster\_file  
    emeans.c, 25

crossover  
    operators.c, 29  
    operators.h, 13

DEBUG\_BOUNDS  
    utility.h, 19

DEBUG\_CENTROIDS  
    utility.h, 19

DEBUG\_CLUSTER  
    utility.h, 19

DEBUG\_CONFIG  
    utility.h, 19

DEBUG\_CROSSOVER  
    utility.h, 20

DEBUG\_DATA  
    utility.h, 19

DEBUG\_DUNN  
    utility.h, 20

DEBUG\_MUTATE  
    utility.h, 20

DEBUG\_PROBABILITY  
    utility.h, 20

DEBUG  
    emeans.c, 26  
    utility.h, 20

data\_cols  
    emeans.c, 25

data\_file  
    emeans.c, 25

data\_rows  
    emeans.c, 26

debug\_code  
    utility.h, 19

dunn\_index  
    fitness.c, 27  
    fitness.h, 12

ERROR  
    utility.h, 20

emeans  
    emeans.c, 24

emeans.c  
    c\_rate, 25  
    centroids\_file, 25  
    cfg, 25  
    cluster\_file, 25  
    DEBUG, 26  
    data\_cols, 25  
    data\_file, 25  
    data\_rows, 26  
    emeans, 24  
    fitness\_file, 26  
    m\_rate, 26  
    main, 24  
    max\_iter, 26  
    n\_clusters, 26  
    opts, 26  
    size, 26  
    trials, 26  
    VERBOSE, 26

error\_code  
    utility.h, 20

fitness.c  
    dunn\_index, 27

fitness.h  
    dunn\_index, 12

fitness\_file  
    emeans.c, 26

- GREEN
  - utility.h, 19
- gen\_probability
  - selection.c, 33
  - selection.h, 17
- inc
  - pcg\_state\_setseq\_64, 7
- include/cluster.h, 9
- include/fitness.h, 11
- include/io.h, 12
- include/operators.h, 13
- include/pcg\_basic.h, 15
- include/selection.h, 17
- include/utility.h, 18
- io.c
  - load\_data, 28
  - save\_results, 28
- io.h
  - load\_data, 12
  - save\_results, 12
- lloyd\_defined
  - cluster.c, 21
  - cluster.h, 10
- lloyd\_random
  - cluster.c, 22
  - cluster.h, 10
- load\_data
  - io.c, 28
  - io.h, 12
- m\_rate
  - emeans.c, 26
- MAGENTA
  - utility.h, 19
- main
  - emeans.c, 24
- max\_iter
  - emeans.c, 26
- mutate
  - operators.c, 29
  - operators.h, 13
- n\_clusters
  - emeans.c, 26
- operators.c
  - crossover, 29
  - mutate, 29
- operators.h
  - crossover, 13
  - mutate, 13
- opts
  - emeans.c, 26
- PCG32\_INITIALIZER
  - pcg\_basic.h, 15
- pcg32\_boundedrand
  - pcg\_basic.c, 31
- pcg32\_boundedrand\_r
  - pcg\_basic.c, 31
- pcg32\_random
  - pcg\_basic.h, 16
- pcg32\_random\_r
  - pcg\_basic.c, 32
- pcg32\_random\_r
  - pcg\_basic.h, 16
- pcg32\_random\_r
  - pcg\_basic.c, 32
- pcg32\_random\_r
  - pcg\_basic.h, 16
- pcg32\_random\_t
  - pcg\_basic.h, 16
- pcg32\_srandom
  - pcg\_basic.c, 32
- pcg32\_srandom
  - pcg\_basic.h, 17
- pcg32\_srandom\_r
  - pcg\_basic.c, 32
- pcg32\_srandom\_r
  - pcg\_basic.h, 17
- pcg\_basic.c
  - pcg32\_boundedrand, 31
  - pcg32\_boundedrand\_r, 31
  - pcg32\_random, 32
  - pcg32\_random\_r, 32
  - pcg32\_srandom, 32
  - pcg32\_srandom\_r, 32
- pcg\_basic.h
  - PCG32\_INITIALIZER, 15
  - pcg32\_boundedrand, 16
  - pcg32\_boundedrand\_r, 16
  - pcg32\_random, 16
  - pcg32\_random\_r, 16
  - pcg32\_random\_t, 16
  - pcg32\_srandom, 17
  - pcg32\_srandom\_r, 17
- pcg\_state\_setseq\_64, 7
  - inc, 7
  - state, 7
- README.md, 20
- RED
  - utility.h, 19
- RESET
  - utility.h, 19
- random\_centroids
  - cluster.c, 22
  - cluster.h, 11
- SUCCESS
  - utility.h, 20
- save\_results
  - io.c, 28
  - io.h, 12
- select\_parent
  - selection.c, 33
  - selection.h, 18
- selection.c
  - gen\_probability, 33
  - select\_parent, 33
- selection.h

- gen\_probability, 17
- select\_parent, 18
- size
  - emeans.c, 26
- src/cluster.c, 20
- src/emeans.c, 23
- src/fitness.c, 27
- src/io.c, 27
- src/operators.c, 29
- src/pcg\_basic.c, 31
- src/selection.c, 33
- state
  - pcg\_state\_setseq\_64, 7
- trials
  - emeans.c, 26
- utility.h
  - DEBUG\_BOUNDS, 19
  - DEBUG\_CENTROIDS, 19
  - DEBUG\_CLUSTER, 19
  - DEBUG\_CONFIG, 19
  - DEBUG\_CROSSOVER, 20
  - DEBUG\_DATA, 19
  - DEBUG\_DUNN, 20
  - DEBUG\_MUTATE, 20
  - DEBUG\_PROBABILITY, 20
  - ERROR, 20
  - SUCCESS, 20
- utility.h
  - BLUE, 19
  - CYAN, 19
  - DEBUG, 20
  - debug\_code, 19
  - error\_code, 20
  - GREEN, 19
  - MAGENTA, 19
  - RED, 19
  - RESET, 19
  - VERBOSE, 20
  - YELLOW, 19
- VERBOSE
  - emeans.c, 26
  - utility.h, 20
- YELLOW
  - utility.h, 19